Randomized Approximation Algorithms for Query Optimization Problems on Two Processors [*]

Eduardo Laber
PUC-RIO
laber@inf.puc-rio.br

Ojas Parekh
Carnegie Mellon University
odp@andrew.cmu.edu

R. Ravi
Carnegie Mellon University
Carnegie Mellon University ravi@andrew.cmu.edu

Abstract

Query optimization problems for expensive predicates have received much attention in the database community. In these situations, the output to the database query is a set of tuples that obey certain conditions, where the conditions may be expensive to evaluate computationally. In the simplest case when the query looks for the set of tuples that simultaneously satisfy two expensive conditions on the tuples and these can be checked in two different distributed processors, the problem reduces to one of ordering the condition evaluations at each processor to minimize the time to output all the tuples that are answers to the query.

We improve upon a previously known deterministic 3-approximation for this problem: In the case when the times to evaluate all conditions at both processors are identical, we give a 2-approximation; In the case of non-uniform evaluation times, we present a $\frac{8}{3}$-approximation that uses randomization. While it was known earlier that no deterministic algorithm (even with exponential running time) can achieve a performance ratio better than 2, we show a corresponding lower bound of $\frac{3}{2}$ for any randomized algorithm.

1 Introduction

The main goal of query optimization in databases is to determine how a query over a database must be processed in order to minimize the user response time. A typical query extracts the tuples in a relational database that satisfy a set of conditions (predicates in database terminology). For example, consider the set of tuples $\{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_1)\}$ and a query which seeks to extract the subset of tuples (a_i, b_j) for which a_i has property $p(a)$ and b_j has property $p(b)$. In many scenarios, the time to evaluate the property $p(x)$ for element x in the tuple can be assumed to be constant (i.e., $O(1)$) and hence the solution involves scanning through all the tuples in turn and checking for the properties. However, in the case of evaluating expensive predicates, when a relation contains complex data as images and tables, this assumption is not necessarily true. In fact, image processing and table manipulation may be very time consuming. In this case, the time spent by those operations must be taken into account when designing a query optimization algorithm. There is some work in this direction [1, 2, 3, 5, 7, 8]. In [7], it is proposed the Cherry Picking (CP) approach which reduces the dynamic query evaluation problem to DBOP (Dynamic Bipartite Ordering Problem), a graph optimization problem, which is the focus of our improvement in this paper.

[*]This work was supported by an NSF-cNPQcollaborative research grant CCR-9900304.

1.1 Problem Statement

An instance of DBOP consists of a weighted bipartite graph $G = (V, E)$ with bipartition (A, B) and a hidden function $\delta : V \to \{0, 1\}$. The weight $w(v)$ of a node $v \in V$ is the estimated time required to evaluate the function δ over v. The goal is to compute as fast as possible the function $\gamma : E \to \{0, 1\}$ defined by $\gamma(u, v) = \delta(u) \times \delta(v)$. The only allowed operation is to evaluate the function δ over a node. In this formulation, note that sets A and B correspond to different attributes of the relation that is queried – the nodes correspond to distinct attribute values and the edges to tuples in the relation. Figure 1(a) shows a graph corresponding to the set of tuples $\{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_1)\}$.

As in [7], we assume a distributed scenario with two available processors: D_1 and D_2 that are used to evaluate the function δ over the nodes of A and B respectively. The goal is to minimize some sort of "makespan," that is, the maximum time by which both D_1 and D_2 finish all their evaluations of δ so as to determine all the answers to the given query. Figure 1(b) shows an instance for DBOP . The value inside each node indicates the value of the hidden function δ. Assume that $w(a) = 1$ for every $a \in A$ and $w(b) = 3$ for every $b \in B$. In this case, a greedy algorithm that evaluates the nodes with largest degree first has makespan 6 since D_1 evaluates a_1 followed by a_2 and a_4, while D_2 evaluates b_1 followed by b_4. Note that after these evaluations, given the shown δ values, we can conclude that we have output all the tuples with $\gamma = 1$ even without having examined many nodes (such as a_3 or b_2). Thus the crux of the problem is to choose dynamically (based on the given w's and the revealed δ values) an order of evaluation of the nodes for the two processors to minimize makespan.

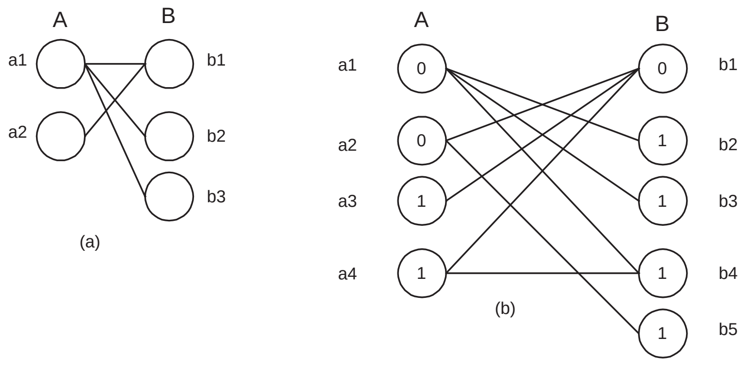

Figure 1: An instance of DBOP

For an ordering algorithm Alg on a DBOP instance I, let $c(Alg, I)$ denote the time for Alg to evaluate γ on I, that is, $c(Alg, I) = \max\{c(Alg, D_1, I), c(Alg, D_2, I)\}$, where $c(Alg, D_1, I)$ and $c(Alg, D_2, I)$ indicate, respectively, the time in which D_1 and D_2 finish their evaluations. In the complexity measure c only the time incurred by the evaluation of δ over the nodes of the graph is charged. When the context is clear we may drop some of the c arguments. We use $t(Alg)$ to denote the total running time spent by Alg on arriving at the evaluation orders in the worst case. Typically, we would like t to be a low-order polynomial in the size of the bipartite graph G.

Define the quality (performance ratio) of an Algorithm Alg for instance I as

$$q(Alg, I) = \frac{c(Alg, I)}{c(Opt, I)},$$

where Opt is the optimal algorithm for the corresponding instance - the one with minimum evaluation time. Furthermore, define the *absolute quality* (quality for short) of an Algorithm Alg as

$$q(Alg) = \max_{I \in Inst} \left\{ \frac{c(Alg, I)}{c(Opt, I)} \right\},$$

where $Inst$ is the set of all possible instances for DBOP .

1.2 Our Results

In [7], it was shown that no deterministic algorithm has quality smaller than 2. A simple algorithm called MBC (Minimum Balanced Cover) was proposed in that paper, which is optimal under the quality metric (has quality 2) but has $t(MBC)$ exponential in the size of G. In order to circumvent this problem, a variant, MBC*, was proposed, that runs in polynomial time, that is $t(MBC^*) = O(|G|^3)$, but with slightly worse quality: $q(MBC^*) = 3$.

In this paper, we first present a simple linear time deterministic algorithm for DBOP which has the best-possible quality of 2 in the case when all weights w are uniform. In the non-uniform case, we present a randomized algorithm that achieves quality $\frac{8}{3}$. Furthermore, we prove that $\frac{3}{2}$ is a lower bound on the expected quality of any (even exponential-time) randomized strategy.

Our paper is organized as follows. In Section 2, we present lower bounds on $c(Opt, I)$. In section 3, we present a deterministic algorithm for uniform weights. In section 4, we present the rMBC algorithm, a randomized version of MBC*. This algorithm is used as a subroutine for 2rMBC , the $\frac{8}{3}$-expected quality algorithm proposed in Section 5. In Section 6, we present a lower bound on the quality of any randomized algorithm for DBOP . Finally, in Section 7 we present our conclusions and indicate some open problems.

2 Lower Bounds

Let $G = (A \cup B, E)$ be a simple bipartite graph with node weights $w(\cdot)$. A (vertex) cover for G is the union of two subsets X and Y, $X \subset A$ and $Y \subset B$, such as that for every edge $e \in E$, at least one of e's endpoints is in $X \cup Y$. Given $V' \subset V$, define $w(V')$ to be $\sum_{v \in V'} w(v)$.

Given a graph G and non-negative numbers g_1 and g_2, the MinMax Cover $MMC(g_1, g_2)$ problem is to find a cover $X \cup Y$ for G that minimizes $\max\{g_1 + w(X), g_2 + w(Y)\}$, among all possible covers for G. This problem is NP-complete as shown in [7]. Furthermore, it admits the following natural integer programming formulation.

$$\begin{array}{rl}
\text{Minimize } t & \\
\text{subject to} & \\
g_1 + \sum_{i \in A} w(a_i) a_i & \leq t \\
g_2 + \sum_{j \in B} w(b_j) b_j & \leq t \\
a_i + b_j \geq 1 & \forall (i,j) \in E \\
a_i \in \{0,1\} & \forall i \in A \\
b_j \in \{0,1\} & \forall j \in B
\end{array}$$

In the above program, a variable a_i (b_j) is assigned to 1 if the node a_i belongs to the cover and it is set to 0, otherwise.

We also consider the well known Minimum Bipartite Vertex Cover (MBVC) problem, which consists of finding a cover $X \cup Y$ for G, with $X \subset A$ and $Y \subset B$, that minimizes $w(X) + w(Y)$. This problem can be solved in $O(|E|(|A| + |B|)\log(|E|))$ time through a max st flow algorithm [4, 6]. We have the following lemmas.

Lemma 1 *Let $I = (G, \delta)$ be an instance of DBOP and let $X^* \cup Y^*$ be a solution of MMC(0,0) for G. Then $c(Opt, I) \geq \max\{w(X^*), w(Y^*)\}$.*

Proof: For every edge $e \in E$, at least one of its endpoints must be evaluated, otherwise it is not possible to determine $\gamma(e)$. Let $X_{opt} \cup Y_{opt}$ be the set of nodes evaluated by Opt when it solves I. Clearly, $X_{opt} \cup Y_{opt}$ is a cover for G, otherwise there is an edge $e \in E$ such that none of its endpoints is evaluated by Opt. Since $X^* \cup Y^*$ is an optimal solution of MMC(0,0) for G, it follows that
$c(opt, I) \geq \max\{w(X_{opt}), w(Y_{opt})\} \geq \max\{w(X^*), w(Y^*)\}$. □

Lemma 2 *Let $I = (G, \delta)$ be an instance of DBOP and let $\overline{X} \cup \overline{Y}$ be a solution of MBVC for G. Then,*

$$c(Opt, I) \geq (w(\overline{X}) + w(\overline{Y}))/2 \tag{1}$$

and

$$c(Opt, I) \geq w(z), \tag{2}$$

for every $z \in \overline{X} \cup \overline{Y}$

Proof: The correctness of (1) follows from lemma 1 and from the fact that

$$(w(\overline{X}) + w(\overline{Y}))/2 \leq \max\{w(X^*), w(Y^*)\},$$

where $X^* \cup Y^*$ be a solution of MMC(0,0) for G.

Now, let z be a node in $\overline{X} \cup \overline{Y}$ and let $N(z)$ be the set of nodes adjacent to z in $A \cup B$. We observe that either Opt evaluates z or every node in $N(z)$, otherwise there is an edge $e \in E$ such that none of its endpoints is evaluated by Opt. Hence, $c(Opt, I) \geq \min\{w(z), w(N(z))\}$. Nevertheless, $\min\{w(z), w(N(z))\} = w(z)$, otherwise replacing z by $N(z)$ we would have a solution for MBVC with weight smaller than that of $\overline{X} \cup \overline{Y}$, which is a contradiction. □

2.1 Further Hybrid Lower Bounds

For the next lower bound, we need additional notation. Let V' be a subset of V. We define $T(V')$ (denoting the "True" nodes of V') by $T(V') = \{v \in V' | \delta(v) = 1\}$. Moreover, we use $N(V')$ to indicate the neighborhood of V'. Finally, we define $NT(V')$ by $N(T(V'))$.

Lemma 3 *Let $I = (G, \delta)$ an instance of DBOP . Then,*

$$c(Opt, I) \geq \max\{w(NT(B)), w(NT(A))\}$$

Proof: Let $a \in NT(B)$. It implies that there is $b \in B$, with $\delta(B) = 1$ such that $(a, b) \in E$. Hence, in order to determine whether $\gamma(a, b) = 1$ or not, the node a must be evaluated. Hence, every node in $NT(B)$ must be evaluated by any algorithm for I. A symmetric argument shows that every node in $NT(A)$ must be evaluated. □

We now present our last lower bound that can be viewed as a generalization of the lower bounds presented in Lemmas 1 and 3. We need to introduce the concept of a essential set.

Definition 1 *A set $V' \subset V$ is a essential set if and only if $v \in NT(V)$ for all $v \in V'$.*

We observe that if V' is a a essential set, then all of its nodes must be evaluated by any algorithm, and in particular, by Opt.

Lemma 4 *Let $I = (G, \delta)$ an instance of DBOP . Furthermore, given two essential sets A' and B', with $A' \subset A$ and $B' \subset B$, let G' be the graph induced by $V \setminus (A' \cup B')$. If $\overline{A} \cup \overline{B}$, with $\overline{A} \subset A$ and $\overline{B} \subset B$, solves MMC(w(A'),w(B')) for the graph G', then*

$$c(Opt, I) \geq \max\{w(A') + w(\overline{A}), w(B') + w(\overline{B})\}$$

Proof: Let $X_{opt} \cup Y_{opt}$, with $X_{opt} \subset A$ and $Y_{opt} \subset B$ be the set of nodes evaluated by Opt when it solves the instance I. Since A' and B' are essential sets, it follows that $A' \subset X_{opt}$ and $B' \subset Y_{opt}$. Furthermore, $(X_{opt} \setminus A') \cup (Y_{opt} \setminus B')$ is a cover for G', otherwise one of the edges in G' cannot be determined. Since $\overline{A} \cup \overline{B}$ solves MMC(w(A'),w(B')) for the graph G',

$$\begin{aligned}\max\{w(A') + w(\overline{A}), w(B') + w(\overline{B})\} &\leq \max\{w(A') + w(X_{opt} \setminus A'), w(B') + w(Y_{opt} \setminus B')\} \\ &= \max\{w(X_{opt}), w(Y_{opt})\} \leq c(Opt, I).\end{aligned}$$

□

3 An Algorithm for Uniform Weights

In this section we present a simple deterministic algorithm which runs in linear time and delivers a quality of 2 in the case when the values of the query weights are identical for all nodes in V.

We assume w.l.o.g that $w(v) = 1$ for all $v \in V$. Let M be some maximal matching of G, and let $M_A = M \cap A$ and $M_B = M \cap B$. Finally we set $K_A = NT(M_B) \setminus M_A$ and $K_B = NT(M_A) \setminus M_B$.

In the first phase of the algorithm we evaluate the function δ in parallel for the nodes in M_A and M_B. This takes time $|M| = |M_A| = |M_B|$. In the second phase we evaluate δ for K_A and K_B in parallel, requring time $\max\{|K_A|, |K_B|\}$. Since $M_A \cup M_B$ is a cover of G, δ is evaluated for at least one endpoint of each edge in the first phase. After the second phase, we have either evaluated both endpoints of an edge, or we have that δ is 0 on one of its endpoints, hence the algorithm is correct.

For the analysis of the quality guarantee we appeal to Lemma 4. Let G' be the graph induced by $V \setminus (K_A \cup K_B)$, and let $\overline{A} \cup \overline{B}$ be an optimizer for $MMC(|K_A|, |K_B|)$ in the graph G' induced by the bipartition $(A \setminus K_A, B \setminus K_B)$. By Lemma 4, we have that $c(Opt, I) \geq \max\{|K_A| + |\overline{A}|, |K_B| + |\overline{B}|\}$. Thus

$$q \leq \frac{|M| + \max\{|K_A|, |K_B|\}}{\max\{|K_A| + |\overline{A}|, |K_B| + |\overline{B}|\}} \leq 2\frac{|M| + \max\{|K_A|, |K_B|\}}{|K_A| + |\overline{A}| + |K_B| + |\overline{B}|} \leq 2\frac{|M| + \max\{|K_A|, |K_B|\}}{|M| + |K_A| + |K_B|} \leq 2,$$

where the penultimate inequality follows from the fact that since G' contains M, and $\overline{A} \cup \overline{B}$ is a cover of G', $|\overline{A}| + |\overline{B}| \geq |M|$.

4 A Randomized Algorithm

In this section we present rMBC , a randomized algorithm for DBOP . This algorithm is used as a subroutine by the 2rMBC algorithm presented in the next section. rMBC can be viewed as a randomized version of the MBC^* algorithm proposed in [7].

Before presenting rMBC , we briefly outline the MBC^* algorithm. MBC^* is divided into three steps. In the first step, the MBVC problem is solved for the input graph G. Let $A^* \cup B^*$, with $A^* \subset A$ and $B^* \subset B$, be the solution for MBVC obtained in the first step. In the second step, the function δ is evaluated for the vertices in $A^* \cup B^*$. At the end of this step, at least one endpoint of every edge has already been computed. Let $e = (a, b)$. If one of the e endpoints, say a, evaluated at the second phase is such that $\delta(a) = 0$, then the evaluation of the other endpoint b is not necessary since $\gamma(e) = 0$. Therefore, only those vertices adjacent to at least one vertex with δ value equal to 1 are evaluated in the third step. The rMBC algorithm is similar to MBC^*, except for the fact that it evaluates the cover nodes following a randomly selected order, while MBC^* follows a fixed order. Figure 2 shows the pseudo-code for rMBC .

From now, we assume w.l.o.g. that $w(A') \geq w(B')$. In this case, the processor D_1 does not become idle during the while loop, and as a consequence,

```
Step 1 : Solve MBVC for the graph $G$ obtaining the sets $A'$ and $B'$.

Do in Parallel
    Processor $D_1$:
        $Ne(A') \leftarrow \emptyset$
        For $i := 1$ to $|A'|$
            Select randomly a node $a$ from $A'$ that has not been evaluated yet
            Evaluate $\delta(a)$
            $Ne(A') \leftarrow Ne(A') \cup NT(a)$
        While there exists a node in $Ne(B')$ that has not been evaluated
        or $D_2$ has not processed all the nodes from $B'$
            Let $a$ be a node from $Ne(B')$ that has not been evaluated yet
            Evaluate $\delta(a)$

    Processor $D_2$ :
        $Ne(B') \leftarrow \emptyset$
        For $i := 1$ to $|B'|$
            Select randomly a node $b$ from $B'$ that has not been evaluated yet
            Evaluate $\delta(b)$
            $Ne(B') \leftarrow Ne(B') \cup NT(b)$
        While there exists a node in $Ne(A')$ that has not been evaluated
        or $D_1$ has not processed all the nodes from $A'$
            Let $b$ be a node from $Ne(A')$ that has not been evaluated yet
            Evaluate $\delta(b)$
```

Figure 2: rMBC Algorithm

$$c(D_1, I) = w(A') + w(NT(B') \setminus A') \tag{3}$$

We start the analyzis of rMBC presenting a technical lemma.

Lemma 5 *Let $\pi(A')$ be a random permutation of the nodes in A' and let $\pi(i)$ be the ith node in this permutation. Moreover, let $\overline{\pi}$ be the reverse permutation of π, that is, $\overline{\pi}(i) = \pi(n-i+1)$, for $i = 1, \ldots, n$. Being $c(D_2, \pi)$ the time in which D_2 ends its task, assuming an execution of rMBC over a permutation π, then*

$$\frac{\max\{c(D_2, \pi), c(D_2, \overline{\pi})\}}{c(Opt, I)} \leq 3 \tag{4}$$

and

$$\frac{c(D_2, \pi) + c(D_2, \overline{\pi})}{c(Opt, I)} \leq 5 \tag{5}$$

Proof: First, we prove (4). We have that

$$\max\{c(D_2, \pi), c(D_2, \overline{\pi})\} \leq w(B') + w(NT(A))$$

On the other hand, it follows from lemmas 2 and 3 that

$$c(Opt, I) \geq \max\{(w(A') + w(B'))/2, w(NT(A))\}$$

The inequality (4) follows from the two previous inequalities.

Now, we consider the inequality (5). Let $\pi_j(A')$ be the set $\{\pi(1), \pi(2), \ldots, \pi(j)\}$ and let $\pi(k)$ be the first node of $\pi(A')$ such that after its evaluation by D_1, the processor D_2 does not become idle anymore. Then, we have that

$$c(D_2, \pi) = w(\pi_k(A')) + w(NT(A') \setminus NT(\pi_{k-1}(A'))), \tag{6}$$

Now, let $\overline{\pi}(\overline{k})$ be the first node of $\overline{\pi}(A')$ such that after its evaluation, the processor D_2 does not become idle anymore. In this case,

$$c(D_2, \overline{\pi}) = w(\overline{\pi}_{\overline{k}}(A')) + w(NT(A') \setminus NT(\overline{\pi}_{\overline{k}-1}(A'))). \tag{7}$$

We divide the analysis into two cases

case 1) $k + \overline{k} \leq |A'| + 1$. In this case we have $w(\pi_k(A')) + w(\overline{\pi}_{\overline{k}}(A')) \leq w(A') + w(\pi(k))$, hence it follows from (6) and (7) that

$$c(D_2, \pi) + c(D_2, \overline{\pi}) \leq w(A') + \max_{a \in A'}\{w(a)\} + 2w(NT(A')). \tag{8}$$

case 2) $k + \overline{k} > |A'| + 1$. In this case we have, $(NT(A') \setminus NT(\pi_{k-1}(A'))) \cap (NT(A') \setminus NT(\overline{\pi}_{\overline{k}-1}(A'))) = \emptyset$. This in addition with the sum of (6) and (7) yields,

$$c(D_2, \pi) + c(D_2, \overline{\pi}) \leq 2w(A') + w(NT(A')). \tag{9}$$

On the other hand, it follows from lemmas 2 and 3 that

$$c(Opt, I) \geq \max\{w(A')/2, \max_{a \in A'}\{w(a)\}, w(NT(A'))\} \tag{10}$$

The inequality (5) can be established through some algebraic manipulations involving (8), (9) and (10). $\square$

Theorem 6 *Let $I = (G, \delta)$ be an instance of DBOP . If*

$$\frac{c(D_1, I)}{c(Opt, I)} \leq 7/3,$$

then

$$E[q(rMBC, I)] \leq 8/3$$

Proof: Let Π be the set of all possible permutations for the nodes in A'. By definition,

$$\begin{aligned} E[q(rMBC, I)] &= \frac{1}{|A'|!} \sum_{\pi \in \Pi} \max\left\{\frac{c(D_1, I)}{c(Opt, I)}, \frac{c(D_2, \pi)}{c(Opt, I)}\right\} \leq \frac{1}{|A'|!} \sum_{\pi \in \Pi} \max\left\{\frac{7}{3}, \frac{c(D_2, \pi)}{c(Opt, I)}\right\} \\ &= \frac{1}{2|A'|!} \sum_{\pi \in \Pi} \left(\max\left\{\frac{7}{3}, \frac{c(D_2, \pi)}{c(Opt, I)}\right\} + \max\left\{\frac{7}{3}, \frac{c(D_2, \overline{\pi})}{c(Opt, I)}\right\}\right) \end{aligned}$$

Hence, in order to establish the result, it suffices to prove that

$$\max\left\{\frac{7}{3}, \frac{c(D_2, \pi)}{c(Opt, I)}\right\} + \max\left\{\frac{7}{3}, \frac{c(D_2, \overline{\pi})}{c(Opt, I)}\right\} \leq 16/3, \tag{11}$$

for every $\pi \in \Pi$. We assume w.l.o.g. that $c(D_2, \pi) \geq c(D_2, \overline{\pi})$. If $c(D_2, \pi) \leq 8/3$, we are done. Hence, we assume that $c(D_2, \pi) > 8/3$. The correctness of (11) can be established by applying the inequality (4) in the cases where $c(D_2, \overline{\pi})/c(Opt, I) \leq 7/3$ and the inequality (5) in the case where $c(D_2, \overline{\pi})/c(Opt, I) > 7/3$. $\square$

5 2rMBC Algorithm

At this section, we present the 2rMBC Algorithm, a polynomial time algorithm with expected quality 8/3.

First, the rMBC algorithm is executed until D_2 finishes evaluating the nodes in B' (we are assuming w.l.o.g. that $w(B') \leq w(A')$). At this point, the execution is interrupted and 2rMBC analyzes the information achieved so far. Based on this analysis, it takes one of the two following decisions: Either to resume the execution of rMBC or to execute a second, new algorithm called 2ndCover . Thus, the algorithm has two phases. The phase 1 is divided into two steps. Let $K = NT(B') \setminus A'$. In step 1, 2rMBC checks if

$$\frac{c(D_1, rMBC, I)}{c(opt, I)} \leq \frac{w(A') + w(K)}{\max\{w(NT(B')), (w(A') + w(B'))/2\}} \leq \frac{7}{3} \tag{12}$$

If the ratio is not larger than 7/3, then it follows from Theorem 6 that the expected quality of 2rMBC (rMBC) for the current instance is not larger than 8/3. In this case, the execution of rMBC is resumed.

If the test in step 1 fails, then step 2 is executed. In this step, 2rMBC computes a new lower bound on $c(Opt, I)$. Let G' the bipartite graph induced by $V \setminus K$. It is easy to check that K is a essential set. It follows from Lemma 4 that if $\overline{A} \cup \overline{B}$ is a solution to the graph G', then $max\{(w(K) + w(\overline{A}), w(\overline{B})\}$ is a lower bound on $c(Opt, I)$. The main problem is that solving MMC may require exponential time. Hence, instead of solving $MMC(w(K), 0)$ on G', 2rMBC solves a linear programming (LP) relaxation of the integer programming formulation for $MMC(w(K), 0)$ presented in Section 2. The LP relaxation is obtained replacing the constraints $a_i \in \{0, 1\}$ and $b_j \in \{0, 1\}$ by $0 \leq a_i \leq 1$ and $0 \leq b_j \leq 1$ respectively. Let (a^*, b^*, t^*) the optimum solution for the LP relaxation. The end of step 2 consists in checking whether

$$\frac{c(D_1, rMBC, I)}{c(Opt, I)} \leq \frac{w(A') + w(K)}{t^*} \leq \frac{7}{3} \tag{13}$$

In the positive case, the execution of rMBC is resumed. Otherwise, the following 2ndCover algorithm is executed.

5.1 The 2ndCover Algorithm

First, the solution of the LP relaxation solved in Step 2 is rounded to obtain a feasible cover $A2 \cup B2$ for G', where $A2 \subset A \setminus K$ and $B2 \subset B$. Let $x = \sum_{i \in A \setminus K} a_i^* w(a_i)$ and $y = \sum_{j \in B} b_j^* w(b_j)$.

The cover is obtained through the following procedure:

```
0) $A2 \leftarrow \emptyset$; $B2 \leftarrow \emptyset$
1) For every $i \in A \setminus K$ such that $a_i^* \geq x/(y+x)$
        $A2 \leftarrow A2 \cup \{a_i\}$
20 For every $j \in B$ such that $b_j^* \geq y/(y+x)$
        $B2 \leftarrow B2 \cup \{b_j\}$
```

Now the processors D_1 and D_2 are used to evaluate the nodes in $A2$ and $B2$ respectively. Finally D_1 and D_2 evaluate the nodes from $NT(B' \cup B2)$ and $NT(A2)$, respectively, that have not yet been processed.

5.2 Algorithm Analysis

In this section, we analyze the performance of 2rMBC .

Theorem 7 *2rMBC is correct.*

Proof: If either inequality (12) or inequality (13) holds, then 2rMBC 's correctness follows from that of rMBC , which in turn follows from the correctness of MBC^* outlined in the introduction of Section 4. On the other hand, if neither inequality (12) nor inequality (13) hold then 2ndCover is executed. Since $A' \cup B'$ is a cover of G, B' must contain the neighbors of K, hence the algorithm processes both endpoints of any edge with an endpoint in K. If $A2 \cup B2$ is a cover of $G' = G[V \setminus K]$ then again an argument analogous to the one used to establish the correctness of MBC^* shows that upon termination of the algorithm, γ may be evaluated for the edges of G'. Thus the only thing which remains to be shown is that $A2 \cup B2$ indeed covers G'. For an edge ij of G', we have that $a_i^* + b_j^* \geq 1 = x/(y+x) + y/(y+x)$, hence at least one of i and j must belong to $A2 \cup B2$. □

Lemma 8 *Let $A2 \cup B2$ be the cover for G' obtained by 2ndCover algorithm. Then,*

$$\max\{w(A2), w(B2)\} \leq x + y$$

Proof: It follows from the rounding scheme that

$$w(A2) = \sum_{i \in A2} w_i \leq \sum_{i \in A \setminus K} a_i^* w_i (y+x)/x \leq \frac{y+x}{x} \sum_{i \in A \setminus K} a_i^* w_i = y + x.$$

The analysis for $B2$ is symmetric. □

Theorem 9

$$E[q(2rMBC))] \leq 8/3$$

Proof: First, we consider the case when the execution of rMBC is resumed after the interruption. In this case, either the test at Step 1 or the test at Step 2 is positive, which implies that

$$\frac{w(A') + w(K)}{c(Opt, I)} \leq \frac{7}{3}.$$

Therefore, it follows from theorem 6 that $E[q(rMBC, I)] \leq 8/3$.

Now, we assume that 2ndCover is executed. In this case, the tests performed at Step 1 and 2 fail, that is, inequalities (12) and (13) do not hold. First, using the observation that inequality (12) fails, we have that

$$w(K) \geq \frac{w(A')}{6} + \frac{7w(B')}{6}. \tag{14}$$

Using the fact that $w(K) + x \leq t^*$, and the fact that (13) does not hold, we have that

$$7x \leq 3w(A') - 4w(K) \tag{15}$$

Replacing the upper bound on $w(K)$ given by inequality (14) in the inequality above, we conclude that

$$x \leq w(A')/3 - 2w(B')/3 \tag{16}$$

We have the following upper bound on $c(2rMBC, I)$.

$$c(2rMBC, I) \leq w(B') + \max\{w(A2), w(B2)\} + \max\{w(NT(A)), w(NT(B))\}.$$

Since $w(A2) \leq x + y$, $w(B2) \leq x + y$ and $y \leq t^*$, we have that

$$c(2rMBC, I) \leq w(B') + x + t^* + \max\{w(NT(A)), w(NT(B))\}, \tag{17}$$

Since $w(NT(A))$, $w(NT(B))$, $(w(A) + w(B))/2$ and t^* are lower bounds on $c(Opt, I)$, then

$$\begin{aligned} q(2rMBC, I) &\leq \frac{w(B') + x + t^* + \max\{w(NT(A)), w(NT(B)\}}{\max\{w(NT(A)), w(NT(B)), t^*, (w(A') + w(B'))/2\}} \\ &\leq 2 + \frac{w(B') + x}{(w(A') + w(B'))/2} \leq 8/3, \end{aligned}$$

where the last inequality follows from (16). □

6 A 3/2 Lower Bound for Randomized Algorithms

In order to give a lower on the expected quality of any randomized algorithm for DBOP , we apply Yao's minmax principle [9].

We consider a distribution D for the instances of DBOP , where the only instances with nonzero probability are all equally likely and have the following properties.

(i) The input graph $G = (V, E)$ has bipartition (A, B), where $A = \{a_1, \ldots, a_{2n}\}$ and $B = \{b_1, \ldots, b_{2n}\}$. Moreover, $E = \{(a_i, b_i) | 1 \leq i \leq 2n\}$ and $w(v) = 1$ for every $v \in V$.

(ii) $|T(A)| = |T(B)| = n$ and $\gamma(e) = 0$ for every $e \in E$.

Observe that there are exactly $\binom{2n}{n}$ instances with these properties. In distribution D, all of them have probability $1/\binom{2n}{n}$. By Yao's principle, the average case quality of an optimal deterministic algorithm for D is a lower bound on the expected quality of any randomized algorithm for DBOP .

Our first observation is that

$$c(Opt, I) = n$$

for every instance I with nonzero probability, once Opt evaluates in parallel the nodes in $A \setminus T(A)$ and $B \setminus T(B)$. Let Opt_D be the optimal deterministic algorithm for distribution D. The expected quality of Opt_D is given by the expected cost of Opt_D divided by n.

In order to evaluate the expected cost of Opt_D, we first calculate the expected cost of the optimal sequential algorithm for distribution D. Next, we argue that the optimal parallel algorithm for D cannot overcome the sequantial one by a factor larger thatn 2. The sequential algorithm uses only one processor to evaluate δ over the nodes of the graph. Hence, its cost is given by the number of evaluated nodes. Let $k_1 \geq k_2$. We use $P(k_1, k_2)$ to denote the expected cost of an optimal sequential algorithm for a distribution D_{k_1,k_2}, where the only instances with nonzero probability are all equally likely and have the following properties:

(i) The input graph $G = (V, E)$ has bipartition (A, B), where $A = \{a_1, \ldots, a_{k_1+k_2}\}$ and $B = \{b_1, \ldots, b_{k_1+k_2}\}$. Moreover, $E = \{(a_i, b_i) | 1 \leq i \leq k_1 + k_2\}$.

(ii) $\max\{A \setminus |T(A)|, B \setminus |T(B)|\} = k_1$, $\min\{A \setminus |T(A)|, B \setminus |T(B)|\} = k_2$ and $\gamma(e) = 0$ for every $e \in E$.

Clearly, the optimal algorithm for distribution D_{k_1,k_2} firstly evaluates a node from the side with k_1 false node. If it succeds to find a false node, say v, then it does not need to evaluate the node adjacent to v. Otherwise, it has. We can formulate the following recursive equation when $k_1 \geq k_2 > 0$.

$$P(k_1, k_2) = \frac{k_1}{k_1 + k_2}[1 + P(\max\{k_1 - 1, k_2\}, \min\{k_1 - 1, k_2\})] + \frac{k_2}{k_1 + k_2}[2 + P(k_1, k_2 - 1)],$$

Moreover, $P(k_1, 0) = k_1$, since the optimal algorithm just need to evaluate the nodes from one side of the graph. It is possible to verify that

$$\lim_{n \to \infty} \frac{P(n, n)}{n} = 3$$

Since, the optimal parallel algorithm Opt_D cannot overcome the sequential one by a factor larger than 2, we have that

$$\lim_{n \to \infty} \frac{E[c(Opt_D)]}{n} \geq 3/2$$

Therefore, we can state the following theorem.

Theorem 10 *Let Alg be a randomized algorithm for DBOP . Then,* $E[q(Alg)] \geq 1.5$.

7 Conclusion

In this paper we have addressed the problem of evaluating queries in the presence of expensive predicates. We presented a randomized polynomial time algorithm with expected quality 8/3 for the case of general weights. For the case of uniform weights, we show that a very simple deterministic linear time algorithm has quality 2, which is the best possible quality achievable by a deterministic algorithm [7]. On the other hand, we gave a lower bound of 1.5 on the expected quality of any randomized algorithm. Two major questions remain open:

1. Is there a polynomial time deterministic algorithm with quality 2 for non uniform weights ?
2. Where exactly in between 1.5 and 2.667 is the exact value of the quality of the best possible polynomial randomized algorithm for DBOP ?

For future research, we intend to perform experiments comparing the results from the randomized algorithm presented in this paper with other available algorithms, in particular, those proposed earlier in [1, 5, 7].

References

[1] L. Bouganim, F. Fabret, F. Porto, and P. Valduriez, *Processing queries with expensive functions and large objects in distributed mediator systems*, in Proc. 17th Intl. Conf. on Data Engineering, April 2-6, 2001, Heidelberg, Germany, 2001, pp. 91–98.

[2] S. Chaudhuri and K. Shim, *Query optimization in the presence of foreign functions*, in Proc. 19th Intl. Conf. on Very Large Data Bases, August 24-27, 1993, Dublin, Ireland, 1993, pp. 529–542.

[3] D. Chimenti, R. Gamboa, and R. Krishnamurthy, *Towards on open architecture for LDL*, in Proc. 15th Intl. Conf. on Very Large Data Bases, August 22-25, 1989, Amsterdam, The Netherlands, 1989, pp. 195–203.

[4] A. V. Goldberg and R. E. Tarjan, *A new approach to the maximum flow problem*, in Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, Berkeley, California, 28–30 May 1986, pp. 136–146.

[5] J. M. Hellerstein and M. Stonebraker, *Predicate migration: Optimizing queries with expensive predicates*, in Proc. 1993 ACM SIGMOD Intl. Conf. on Management of Data, May 26-28, 1993, Washington, D.C., USA, 1993, pp. 267–276.

[6] D. S. Hochbaum, *Approximating clique and biclique problems*, Journal of Algorithms, 29 (1998), pp. 174–200.

[7] E. S. Laber, F. Porto, P. Valduriez, and R. Guarino, *Improving the complexity of warm-up algorithm*, Tech. Rep. 01, Departamento de Informática, PUC-RJ, Rio de Janeiro, Brasil, April 2002.

[8] T. Mayr and P. Seshadri, *Client-site query extensions*, in Proc. 1999 ACM SIGMOD Intl. Conf. on Management of Data, June 1-3,1999, Philadelphia, Pennsylvania, USA, 1999, pp. 347–358.

[9] A. C. Yao, *Probabilistic computations : Toward a unified measure of complexity*, in 18th Annual Symposium on Foundations of Computer Science, Long Beach, Ca., USA, Oct. 1977, IEEE Computer Society Press, pp. 222–227.